CS 547 Lecture 2: Queueing and Fundamental Laws

Daniel Myers

System Terminology

A *system* is a collection of *resources*. *Customers* enter the system and follow a *routing* through the resources. Routings can be fixed (every customer visits the same set of resources in the same order), or probabilistic (customers may take randomly chosen paths through the system).

There are two main types of systems: *open* and *closed*.

- In an open system, customers arrive from the outside world according to some process, travel through the system, then exit back to the outside world and do not return.
- In a closed system, a fixed population of customers circulates continuously throught the system. There are no outside arrivals or departures.

Queueing

The majority of our analysis will deal with resources that experience contention. The simplest such model is the *single-server queue*. This resource has a single waiting line and a single service center. A customer arrives at the queue, waits for its turn to be served, receives its required amount of service, then departs.

A queue is a statistical model. Its arrivals are governed by some kind of stochastic process (usually described in terms of the statistical properties of the interarrival times), and the required service times are generally random with a particular statistical distribution. The behavior of the queue is determined by the interaction of arrivals and service times, along with scheduling behavior and other fixed characteristics.

The most important queueing parameters are

- $\overline{s}$, the average service time
- $\mu = \frac{1}{\overline{s}}$, the average service rate, which is useful in some contexts
- σ_s^2, the variance in the service times
- λ, the arrival rate

We will always assume that these parameters are stable over the modeling interval.

Using these basic parameters, we can calculate the following derived parameters:

- $\overline{W}$, the average waiting time
- $\overline{R} = \overline{W} + \overline{s}$, the average residence time of a customer in the queue
- $\overline{Q}$, the average number of customers in the queue, including the one in service
- U, the utilization, or fraction of time the server is busy serving a customer

By default, queues use first-come-first-serve (FCFS) scheduling. Later, we will examine other scheduling strategies and see what effect they have on performance.

Throughput

Define Λ to be the *system throughput*, the rate at which customers depart from the system.

Over the long term, $\Lambda \leq \mu$. Every customer requires service, so we cannot complete customers faster than they can be served. In addition, $\Lambda \leq \lambda$, because customers cannot be completed faster than they arrive.

If $\Lambda = \lambda$, the system is *stable*. If $\Lambda < \lambda$, the system is *overloaded*. Due to variability in the arrival process, there will always be short bursts where $\Lambda < \lambda$, but these even out over time. We will assume that our queues always have enough buffer space to hold any customers that arrive during period of temporary overload. If the system is permanently overloaded, the number of customers in the system grows to infinity.

Throughput and response time are not the same thing! They are related, but fundamentally different, measures of system performance. In general, users care about response time – they want to minimize the time required to get a response. System owners generally care about throughput – they want to serve as many customers as possible for the lowest possible cost.

Conservation Law

What goes in must (normally) come out.

Consider a system with arrival rate of λ and an output rate of Λ. If the system is not overloaded and no customers are created or destroyed inside the system, then $\lambda = \Lambda$.

Creating customers in a system is called *forking*. Destroying customers is called *joining*. It's possible to model systems with these behaviors, but usually difficult, so we won't see any examples until the end of class.

Note that this law applies to any system, not just a single queue.

The Forced-Flow Law

$\Lambda_k = \overline{V}_k \Lambda$

The Forced-Flow Law (FFL) relates throughputs at individual resources within a system to the overall system throughput.

Suppose the overall system has arrival rate λ and output rate Λ, and the k^{th} resource has input rate λ_k and output rate Λ_k.

Let $\overline{V}_k$ be the average *visit count* of resource k, representing the average number of times each customer in the system arrives to resource k. If $\overline{V}_k > 1$, then customers, on average, visit resource k multiple times. If $\overline{V}_k < 1$, then only some customers visit resource k and others never visit it.

The Forced-Flow Law:

$$\lambda_k = \overline{V}_k \lambda.$$

The average arrival rate to resource k is the total system arrival rate times the expected number of visits made to resource k. Because of the Conservation Law, we could also state the FFL in terms of output rates, Λ and Λ_k.

Proof

The FFL is so intuitive it almost doesn't require a proof. Nonetheless, here is a measurement-based argument for its validity. Suppose we measure a system over a period of time T and record C completions at the system as whole and C_k competions at resource k. The following relationships hold:

$$\Lambda = \frac{C}{T} \quad \Lambda_k = \frac{C_k}{T} \quad \overline{V}_k = \frac{C_k}{C}$$

Assembling everything together, we have

$$\frac{C_k}{T} = \frac{C_k}{C} \cdot \frac{C}{T}$$

$$\Lambda_k = \overline{V}_k \Lambda.$$

This type of measurement-based reasoning is arguably not a "real" proof because we've been very vague about the nature of the interval under measurement or how we defined completions. Nonetheless, this type of thinking is potentially more useful than a rigorous proof for understanding system dynamics.

Forced-Flow Law Example

We're given a video server with 8 disks that serves 5 clients per minute. Each client requests one 30 minute video, which is streamed at 300 kb/s. The videos are stored on the disks in 4KB blocks, with the blocks for each video equally spread across all the disks.

What is the average number of visits to each disk per client?

We need to calculate $\overline{V}_{disk}$, which is a function of the number of disks and the number of 4KB blocks read by each client request.

$$\overline{V}_{disk} = \frac{\text{number of blocks per video}}{8}$$

The number of blocks per video is a function of the size of the video and the block size.

$$\text{bits per video} = 30\frac{min}{video} \cdot 60\frac{sec}{min} \cdot 300\frac{kb}{sec}$$

$$\text{bits ber block} = 4096\frac{bytes}{block} \cdot 8\frac{bits}{byte}$$

Performing the calculation yields 16875 blocks per video. Dividing this value by 8 to obtain the visits to each disk gives $\overline{V}_{disk} \approx 2100$ accesses/video.

We can use the FFL to estimate the throughput at the disk.

$$\Lambda = 2100\frac{accesses}{video} \cdot 5\frac{videos}{min} \cdot \frac{1}{60}\frac{min}{sec}$$

$$= 175\frac{accesses}{sec}$$

Matching this throughput would require $\overline{s}_{disk} < 6$ ms, which may be unrealistically fast. Adding more disks would reduce the per-disk throughput and allow a larger average service time.